**API testiranje - automatizacija**

Uvod

Testiranje API-ja je ključni deo razvoja softvera. API-ji služe kao kanali komunikacije između različitih komponenti softvera, omogućavajući im da međusobno komuniciraju i razmenjuju podatke. Testiranje API-ja osigurava da ovi interfejsi funkcionišu ispravno, izvršavaju se kako se očekuje i pružaju željene rezultate.

Testiranje API-ja ima nekoliko važnih ciljeva:

* Funkcionalnost: Provera da li API radi ispravno i da li pravilno obrađuje zahteve korisnika. Ovo uključuje testiranje različitih scenarija, uključujući ispravne i neispravne zahteve, obradu grešaka, autentifikaciju i autorizaciju.
* Pouzdanost: Provera da li API pravilno odgovara na zahteve i vraća očekivane rezultate. Ovo uključuje testiranje odgovora na greške, rukovanje sa neočekivanim situacijama i očekivanje stabilnosti i doslednosti u radu API-ja.
* Performanse: Provera performansi API-ja u smislu vremena odziva, opterećenja i skalabilnosti. Testiranje performansi pomaže identifikaciji eventualnih uskih grla i optimizaciji API-ja kako bi se postigao optimalan rad i efikasnost.
* Sigurnost: Provera sigurnosnih aspekata API-ja, kao što su autentifikacija, autorizacija, zaštita od napada poput injekcija, XSS napada i CSRF napada. Testiranje sigurnosti pomaže u otkrivanju i ispravljanju ranjivosti i obezbeđuje da API bude zaštićen od potencijalnih pretnji.
* Integracija: Provera kako API funkcioniše u okviru šireg sistema i kako se povezuje sa drugim komponentama. Testiranje integracije pomaže identifikaciji eventualnih problema u međusobnoj komunikaciji između API-ja i drugih delova sistema.

Automatizacija testiranja API-ja donosi brojne prednosti, kao što su poboljšana efikasnost, brže povratne informacije, veći opseg testiranja i sposobnost rukovanja složenim scenarijima. Omogućava ponovno izvršavanje testnih slučajeva i smanjuje ručni napor potreban za testiranje regresije. Automatizacija takođe olakšava rano otkrivanje problema, omogućava brže ispravljanje grešaka i sprečava nagomilavanje tehničkog duga.

Razvio sam test framework kako bi testirao API endpointe web stranice <https://automationexercise.com/.> Framework pruža skup test slučajeva koji pokrivaju različite aspekte API-ja, omogućavajući testiranje i validaciju njegove funkcionalnosti.

Framework za automatizaciju testiranja API-ja pruža nekoliko prednosti u odnosu na manuelno testiranje. Omogućava testerima da brzo i tačno izvrše ponavljajuće i kompleksne testne scenarije, što dovodi do značajne uštede vremena i povećane produktivnosti. Osim toga, automatizovano testiranje smanjuje rizik od ljudskih grešaka, obezbeđujući dosledne i pouzdane rezultate.

Korišćene tehnologije i alati

Java

Java je moćan programski jezik koji se često koristi u razvoju softvera, a posebno se ističe u podršci za testiranje. Sa širokim izborom test frameworka koji su dostupni za Javu, programeri imaju mogućnost da efikasno pišu, izvršavaju i održavaju testove svojih aplikacija.

JUnit

Jedan od najpoznatijih frameworka za testiranje je JUnit. JUnit omogućava pisanje jednostavnih i strukturiranih testova jedinica (unit testova) koji proveravaju funkcionalnost pojedinačnih delova koda. Ovaj framework pruža bogat set asertivnih metoda za proveru očekivanih rezultata i omogućava lako automatizovano izvršavanje testova.

Apache Maven

Apache Maven je popularan alat za upravljanje projektima i izgradnju softvera u Javi. Maven koristi XML bazirane konfiguracione fajlove, poznate kao POM (Project Object Model), koji omogućavaju programerima da definišu detalje projekta i zavisnosti. Maven pojednostavljuje proces izvršavanja testova pružajući standardizovanu strukturu projekta i upravljanje životnim ciklusom.

Rest-Assured

Rest-Assured je popularna Java biblioteka za testiranje RESTful API-ja. Ona pruža jednostavne metode za izvršavanje HTTP zahteva, proveru statusnih kodova, proveru odgovora, kao i validaciju JSON ili XML odgovora. Rest-Assured olakšava pisanje automatizovanih testova za proveru funkcionalnosti API-ja, omogućavajući programerima da lako izvrše zahteve, provere rezultate i upravljaju sa sekcijama odgovora. Sa svojom intuitivnom sintaksom i bogatim skupom funkcionalnosti, Rest-Assured je moćan alat koji olakšava testiranje RESTful servisa u Java razvojnom okruženju.

Java Faker

Java Faker je praktična Java biblioteka koja se koristi za generisanje lažnih podataka u svrhe testiranja i razvoja. Ona pruža širok spektar metoda za kreiranje realističnih podataka kao što su imena, adrese, brojevi telefona, datumi i drugo. Sa Java Fakerom, programeri mogu jednostavno generisati slučajne podatke koji liče na stvarne, što je korisno za popunjavanje baza podataka ili simuliranje korisničkih interakcija u testnim okruženjima. Uključivanjem Java Fakera u svoje projekte, programeri mogu uštedeti vreme i napor pri ručnom generisanju testnih podataka, poboljšavajući efikasnost procesa razvoja softvera.

Objašnjenje koda

pom.xml

Na osnovu pom.xml fajla Maven ubacuje potrebne biblioteke u projekat. Prikazan je isečak koda iz pom.xml fajla gde se prikazuje kako se formiraju zavisnosti (dependencies) na osnovu čega Maven zna šta treba da ubaci u projekat.

<dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.12</version>  
 </dependency>  
 <dependency>  
 <groupId>io.rest-assured</groupId>  
 <artifactId>rest-assured</artifactId>  
 <version>4.4.0</version>  
 </dependency>  
 <dependency>  
 <groupId>com.github.javafaker</groupId>  
 <artifactId>javafaker</artifactId>  
 <version>1.0.2</version>  
 </dependency>  
</dependencies>

Test klasa

@Test  
public void createAccount() {  
 HashMap<String, String> user = createUser();  
  
 *given*()  
 .baseUri(*BASE\_URL*)  
 .formParams(user)  
 .when()  
 .post("/api/createAccount")  
 .then()  
 .statusCode(200)  
 .extract().response().prettyPrint();  
}

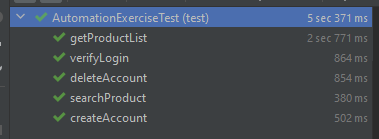
Prikazan je primer jednog testa. Anotacija @Test dolazi iz JUnit biblioteke i na osnovu toga program zna da se radi o testu. Prvo se napravi heš mapa sa random generisanim podacima pomoću Java Faker-a o jednom korisniku. Given metoda i ostale metode koje se vežu za nju dolazi iz Rest-Assured biblioteke koja radi proces HTTP zahteva i proverava da li smo dobili željeni odgovor. U ovom slučaju spakuju se korisnički podaci u formu i šalje se POST zahtev na dati endpoint. Poslednja linija koda ispisue dobijeni odgovor u konzolu.

Metoda koja generiše random podatke o korisniku:

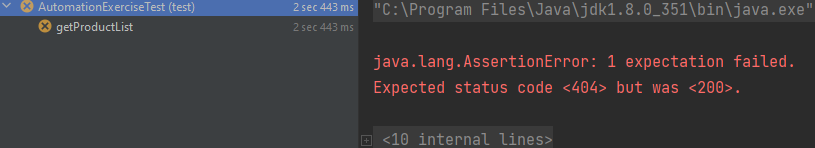
public HashMap<String, String> createUser(){  
 Faker faker = new Faker();  
 HashMap<String, String> user = new HashMap<>();  
 String username = faker.name().username();  
  
 user.put("name", username);  
 user.put("email", username + faker.number().numberBetween(10000, 999999) + "@test.com");  
 user.put("password", "fishtest123");  
 user.put("title", "Mr");  
 user.put("birth\_date", faker.number().numberBetween(1, 28) + "");  
 user.put("birth\_month", faker.number().numberBetween(1, 12) + "");  
 user.put("birth\_year", faker.number().numberBetween(1970, 2007) + "");  
 user.put("firstname", faker.name().firstName());  
 user.put("lastname", faker.name().lastName());  
 user.put("company", faker.company().name());  
 user.put("address1", faker.address().streetAddress());  
 user.put("address2", "");  
 user.put("country", faker.address().country());  
 user.put("zipcode", faker.address().zipCode());  
 user.put("state", faker.address().state());  
 user.put("city", faker.address().city());  
 user.put("mobile\_number", faker.phoneNumber() + "");  
  
 System.*out*.println("username: " + username);  
 System.*out*.println("email: " + user.get("email"));  
  
 return user;  
}

Analiza rezultata

Kada se pokrene test klasa izvrše se svi testovi. Takođe moguće je pokrenuti testove pojedinačno. Na prikazanom screenshot-u se vidi da su svi testovi uspešno prošli.



Napravio sam scenario gde će test da fejluje da se prikaže taj slučaj. U konzoli možemo videti razlog greške. Očekivao sam statusni kod 404 ali sam dobio 200.



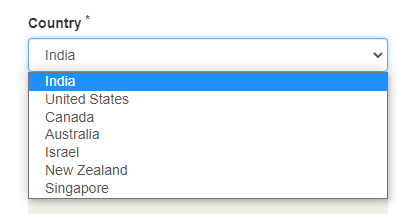
Zaključak

Najbolji način da se iztestira API je da se upoređuju podaci iz baze podataka sa podacima koje vraća API. Ja nažalost nisam imao pristup bazi pa to nisam mogao uraditi. Napravio sam tesove koji proveravaju statusne kodove.

Prilikom testiranja API-a od <https://automationexercise.com/> sajta došao sam do zaključka da im API nije baš najbolji. Počevši od toga da im API uvek vraća statusni kod 200 što znači de je sve OK. Pravi statusni kod možemo videti u response body-ju.

Takođe UI i API se ne podudaraju. Primer ovoga je da kada neko želi da se regisruje ima samo par opcija za County ali sa API zahtevom možemo bilo koji tekst uneti u to polje.

UI:



Podaci kreiranog korisnika sa podatkom koji se ne nalazi u select listi:

